AL/HR-TP-1996-0011

EVALUATION OF MODULAR SEMI-AUTOMATED FORCE AIR ENTITY SIMULATION

Michael T. Conquest
David J. Lerman

Hughes Training, Inc. - Training Operations
6001 South Power Road, Building 561
Mesa, Arizona 85206-0904

Herbert H. Bell

HUMAN RESOURCES DIRECTORATE
AIRCREW TRAINING RESEARCH DIVISION
6001 South Power Road, Building 558
Mesa, Arizona 85206-0904

September 1996

Final Technical Paper for Period November 1995 - January 1996

ARMSTRONG LABORATORY

AIR FORCE MATERIEL COMMAND
BROOKS AIR FORCE BASE, TEXAS

NOTICES

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Office of Public Affairs has reviewed this report, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This report has been reviewed and is approved for publication.

Herbert H. Bell

HERBERT H. BELL
Project Scientist

Elizabeth L. Martin

ELIZABETH L. MARTIN
Technical Director

LYNN A. CARROLL, Colonel, USAF
Chief, Aircrew Training Research Division

Please notify AL/HRPP, 7909 Lindbergh Drive, Brooks AFB, TX 78235-5352, if your address changes, or if you no longer want to receive our technical reports. You may write or call the STINFO Office at DSN 240-3853 or commercial (210) 536-3853.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1996	Final - November 1995 to January 1996

4. TITLE AND SUBTITLE

Evaluation of the Modular Semi-Automated Force Air Entity Simulation

5. FUNDING NUMBERS

C - F41624-95-C-5011
PE - 63227F
PR - 2743
TA - 25
WU - 06

6. AUTHOR(S)
Michael T. Conquest
David J. Lerman
Herbert H. Bell

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Hughes Training, Inc.
Training Operations
6001 South Power Road, Building 561
Mesa, AZ 85206-0904

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Armstrong Laboratory
Human Resources Directorate
Aircrew Training Research Division
6001 South Power Road, Building 558
Mesa, AZ 85206-0904

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AL/HR-TP-1996-0011

11. SUPPLEMENTARY NOTES

Armstrong Laboratory Technical Monitor: Dr. Herbert H. Bell (602) 988-6561

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report describes the Modular Semi-Automated Force (ModSAF) system's capability to provide realistic computer-generated aircraft for use in Advanced Distributed Simulation exercises. This evaluation looked at selected flight performance and mission behavior as well as aerodynamic and flight dynamic exercises. Results indicate that ModSAF provides only a rudimentary capability to realistically depict modern fighter aircraft. Although the ModSAF software architecture is sound and well organized, incorrect modeling of aircraft and mission behaviors presents serious limitations in using ModSAF to represent current Air Force weapons systems. Recommendations include a complete redesign of the fixed-wing aircraft algorithms and data files as well as an expanded set of mission behaviors.

14. SUBJECT TERMS	15. NUMBER OF PAGES
ADS; Advanced distributed simulation; Aerodynamics; Air entities; CGFF; Computer-generated forces; Exercises; Flight dynamics; Flight simulation; Flight simulators; Flight trainers; Mission behaviors; Modeling; ModSAF; Modular Semi-Automated Forces	28
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

CONTENTS

LIST OF FIGURES

PREFACE

This work was conducted by Hughes Training, Inc., Training Operations (HTI) under a contract with Armstrong Laboratory, Human Resources Directorate, Aircrew Training Research Division (AL/HRA), which is located at Williams Gateway Airport in Mesa, Arizona. This effort was undertaken at the request of the Electronic Systems Center at Hanscom Air Force Base, Massachusetts.

The effort was performed under AL/HRA Contract Number F41624-95-C-5011 and Work Unit Number 2743-25-06, Aircrew Training Research Support. The laboratory contract monitor was Mr Daniel Mudd; the laboratory technical monitor was Dr Herbert H. Bell.

EVALUATION OF MODULAR SEMI-AUTOMATED FORCE AIR ENTITY SIMULATION

OBJECTIVES

This evaluation was conducted at the request of the Electronic Systems Center, Hanscom AFB, MA. Its purpose was to assess Modular Semi-Automated Force (ModSAF) capabilities as a computer-generated force (CGF) system for realistically depicting airborne weapon systems in Advanced Distributed Simulation exercises. Both simulated mission tasks and system analyses were used to determine the weapon system's flight performance characteristics, doctrinal behaviors, and overall ModSAF system performance. No attempt was made to compare ModSAF capabilities to other CGF systems.

AIR ENTITIES

ModSAF 2.0 has a very limited number of aircraft types modeled. This evaluation focused on the F-16D because it is the only USAF fighter aircraft modeled in ModSAF 2.0. In addition, F-16 pilots, simulation engineers, and two simulator-certified F-16C Weapon System Trainers (WST) were available to support this evaluation because the Armstrong Laboratory, Aircrew Training Research Division, conducts training research and development using F-16 aircrew training devices.

Although the F-16D was selected, the ModSAF depiction was inadequate to complete all evaluation areas. First, the "D" model is not a particularly good representative of the F-16 for this purpose. The F-16D is a two-seater primarily used for F-16C training. With the exception of F-16 training wings, there are only a few F-16Ds at each base operating F-16Cs. The F-16C would be a better model to represent the F-16 aircraft in ModSAF. Next, the F-16 is a multi-role fighter, capable of air-to-air and air-to-surface missions. ModSAF limits the F-16D to the air-to-surface role. The weapon load selections available for the ModSAF F-16D are inadequate and, in some cases, incorrect. ModSAF correctly allows you to load AGM-65 (Maverick), AIM-9 (Sidewinder), and Mk-82 bombs. ModSAF also allows you to load 2,000 M-50 series 20 mm gun rounds. This is incorrect since the F-16 has a 20 mm M61A1 cannon capable of carrying 512 rounds. Also, ModSAF should include other weapons, such as AIM-120 (AMRAAM), area munitions such as CBU-87, Mk-84 bombs, and laser-guided bombs such as GBU-10 and GBU-12. In addition, ModSAF does not distinguish between the different variants of the same munition, e.g., AGM-65A versus AGM-65D nor different variants of the same aircraft, e.g., F-16C/D Block 30 versus Block 40.

Because the ModSAF F-16D is limited to air-to-ground behaviors, the ModSAF F-14D was used to evaluate air-to-air task behaviors. In addition, the MiG-29 was used for tests requiring an opposing force (OPFOR).

TESTS

All tests were conducted using ModSAF 2.0 as distributed without modification. ModSAF Kits A and C distribution documentation as well as on-line documentation were used to build and run the test scenarios. The software was run on two Silicon Graphics Inc. (SGI) workstations. One Indigo 2 Extreme ran as the SAF simulator while an Indy served as the SAF station. All models were completely dependent on ModSAF 2.0 software. All aircraft behaviors were controlled through the use of ModSAF Task Frames with no additional intelligent forces interface such as SOAR (taking a State, applying an Operator, And generating a Result). Additional test tools included the F-16C WSTs and a Sun 4 workstation for network data collection and analysis. Distributed Interactive Simulation (DIS) protocol 2.04 was used for all network-based testing.

Tests were designed to determine the fidelity of the ModSAF F-16D's aerodynamic and behavioral representation. Climb, turn, and acceleration tests were developed to determine aerodynamic fidelity. Scenarios were also built to test the threat detection, bingo fuel, Winchester weapons, and terrain-following capabilities of ModSAF. The following sections describe these tests and the results.

Climb Performance

Tests were conducted to determine the F-16D's climb performance with different weapon system configurations. After initial results showed that the desired outcome could not be achieved, other tests were added replacing the F-16 thrust map with the default F-16D parameters which surprisingly include the F-14D thrust map.

Figure 1 shows the climb performance test scenario. The test began with the F-16D flying an ingress route at .87 Mach, 3,000 ft MSL. Upon completion of the initial leg of the ingress, the F-16D was tasked to perform an immediate climb to 33,000 ft. For these tests, all weapons were removed and the fuel set to 6,294 lbs.

F-16 Thrust Map Climb Performance

This test scenario was initially run using the F-16 thrust map in the fixed-wing configuration menu. All other configurations remained at the default settings. The F-16D could reach only 4,541 ft in altitude. A closer look at the data revealed that the aircraft rotated at 290 kts, the default take-off speed, and could not accelerate to the commanded ingress speed of .87 Mach (560 kts). When the aircraft attempted to climb for the second phase of the ingress, it eventually stalled and crashed.

A second attempt was made, increasing the fixed-wing aircraft configuration take-off speed to 560 kts, same as the commanded ingress speed. The F-16D was able to achieve 19,554 ft altitude but again stalled after losing all airspeed.

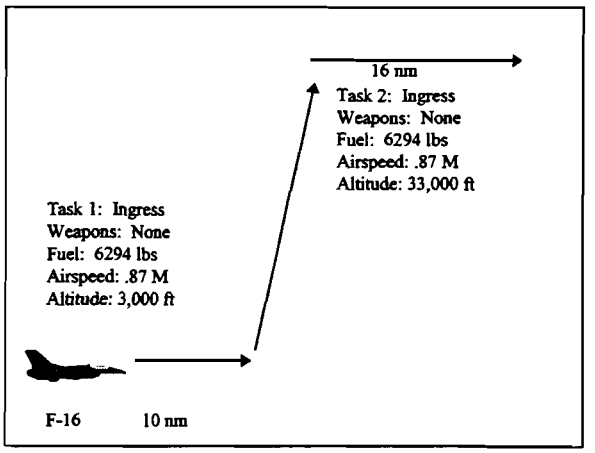

Figure 1
Climb Performance Task

The test scenario was modified a number of times, moving the ingress tasks apart to command a slower climb and using various take-off speeds, all with similar results. Each time, the aircraft attempted to maintain a steep climb angle until it stalled. Then it rapidly went into a steep dive and could not recover.

F-14 Thrust Map Climb Performance

Since the initial test using the F-16 thrust map showed the ModSAF F-16D unable to complete the test, we repeated the test using the F-14 Thrust Map. The F-14 thrust map is the default for the F-16D entity. No documentation could be found telling the user to change the thrust map configuration when creating an F-16D.

The F-16D climbed beyond the commanded 33,000 ft to over 45,000 ft. It achieved 33,000 ft in approximately 30 seconds. With the F-14 thrust map, the ModSAF F-16D climbed much faster and maintained airspeed longer in the climb than an F-16 aircraft.

F-16C Weapon System Trainer Climb Performance

To verify that the ModSAF F-16D, using either of the thrust maps, produces unrealistic flight performance, Armstrong Laboratory's F-16C Weapon System Trainer was configured the same as the ModSAF F-16D, and the climb performance was recorded. The F-16C WST was able to achieve 33,000 ft in approximately 60 seconds. The F-16C WST reached 33,000 ft within the distance between the end of the first ingress phase and the beginning of the second ingress. However, the F-16C could not maintain the steep flight path angle attempted by the ModSAF F-16 test. The F-16C WST gradually decreased the flight path angle to maintain its airspeed during the climb. As previously noted, the ModSAF F-16s maintained the steep flight path angle until they stalled. Figure 2 illustrates the climb performance of the ModSAF F-16D with both thrust maps and the F-16C WST.

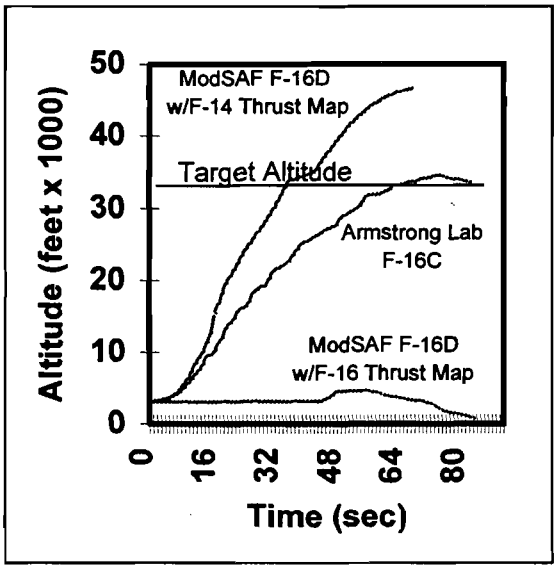

Figure 2
F-16 Climb Performance Comparison

Effects of Weapons/Fuel Load on Performance

Additional climb performance tests were conducted to determine the effects of weapons load and fuel on the ModSAF F-16D flight performance. The previous climb test scenarios were used but with the take-off speed modified to 560 kts to achieve the same initial speed during all tests when entering the climb. The F-16 thrust maps were used on all three tests.

The first run was conducted with the F-16D configured with no weapons and 6,294 lbs of fuel. It was able to achieve 19,554 ft of altitude before stalling. The second configuration increased the fuel load to 12,884 lbs with no weapons. In this configuration, the F-16D could reach only 17,035 ft before stalling. Finally, we added weapons: 4 Mavericks, 2 Sidewinders, 6 Mk-82s, and 2,000 M50s with the 12,884 lb fuel load. The F-16 reached 17,039 ft in this configuration.

Figure 3 shows that the ModSAF F-16D's fuel load does impact climb performance. It also shows that the ModSAF did not model the additional drag and weight that weapons add to the aircraft.

Turn/Acceleration Testing

Test scenarios were designed to determine turn and acceleration performance. However, the incorrect thrust modeling caused the turn rate tests to be inconclusive. The turn data collected could not determine whether the turn performance was correctly modeled. While

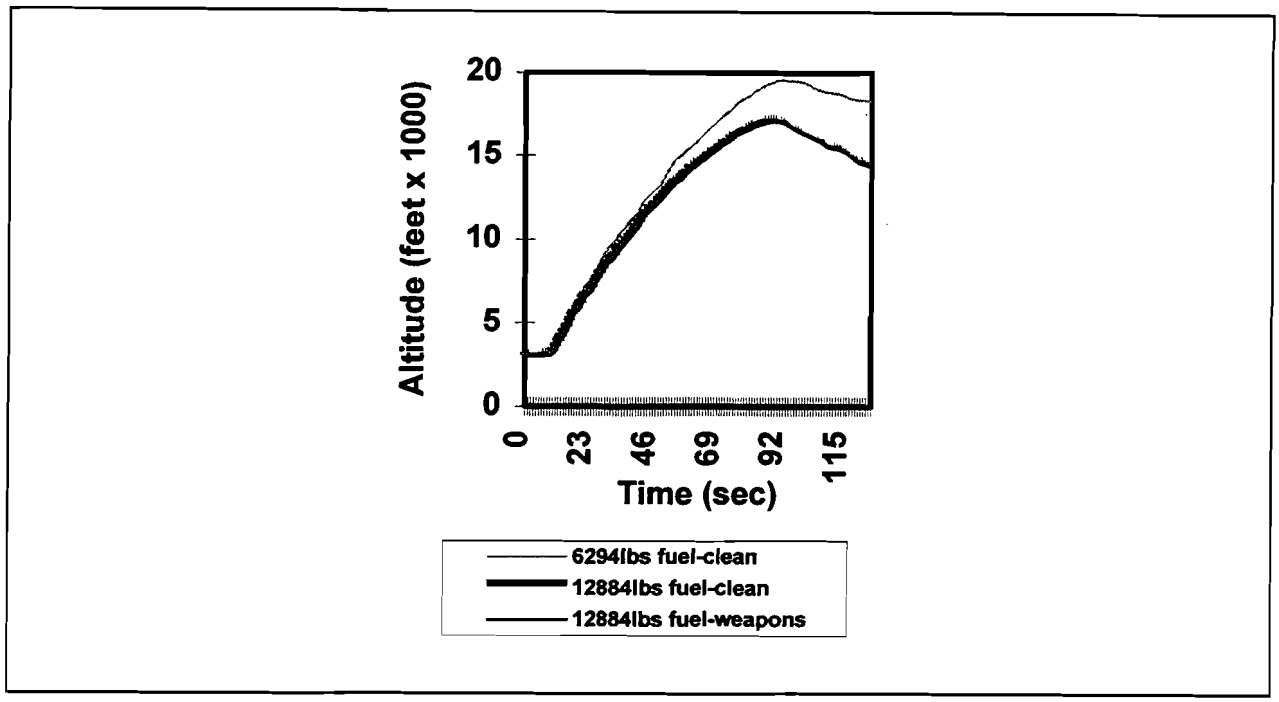

Figure 3
Climb Performance Variation Due to Configuration

using the F-16 thrust map, target speeds could not be obtained for comparison to the flight manual data. It was also difficult to determine if the ModSAF aircraft was attempting a maximum rate turn, a standard rate turn, or something in between. Acceleration tests were not conducted because the data collected during climb performance clearly indicated incorrect thrust or drag modeling.

Threat Detection - Air

A ModSAF sweep scenario with the MiG-29 was created, networked to the manned F-16 WST for testing the air-to-air threat detection capabilities of ModSAF. The MiG-29 was placed on a straight sweep route with the radar on +/- 40 deg azimuth. The ModSAF status window was used to determine when the MiG-29 was aware of the F-16.

The F-16 WST was initialized approximately 10 nm in trail of the MiG-29. The F-16 locked on the MiG-29 with radar and continued to close without detection by the MiG-29. This result was consistent with the ModSAF documentation of the detection model but is not realistic and reflects the absence of a standardized emission Protocol Data Unit (PDU) in the Distributed Interactive Simulation communication protocol. Such a standardized emission PDU is required before ModSAF can effectively incorporate a radar warning receiver detection model.

The F-16 WST then moved wing-to-wing with the MiG-29. Even though the F-16 was well within visual range, the MiG-29 did not detect the F-16. The status page would intermittently change to "avoiding collision with nearby aircraft," but never detected the F-16 as a threat. This verified the ModSAF documentation that indicates the ModSAF does not consider visual threat detection during air-to-air engagements.

Once the F-16 overtook the MiG-29 and was within its radar volume, the MiG-29 detected the threat. The F-16 continued approximately 10 nm ahead of the MiG-29 and then turned back toward the MiG-29. The MiG-29 then began targeting and firing on the F-16 consistent with the user-entered rules of engagement and commit criteria.

Threat Detection - Ground

Figure 4 shows the test scenario used to determine how the ModSAF F-16D would react to ground threats along a pre-planned route. Two flights of F-16Ds were given a low-level ingress route to a target area. An SA-9 platoon was placed along the route. The F-16D flights were separated by approximately 30 seconds to determine if any communication or command and control is modeled that would allow the second flight to alter their behavior based on information obtained from the lead flight.

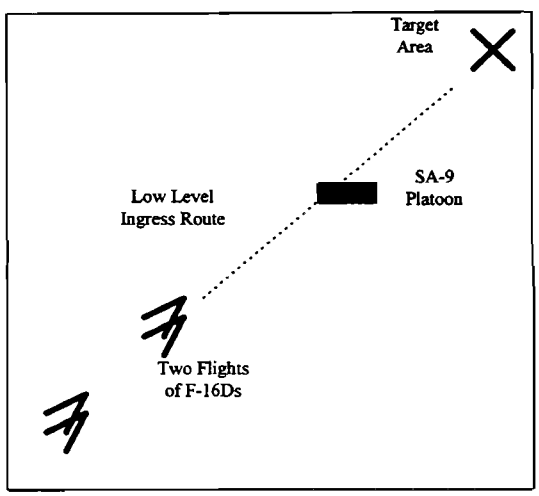

Figure 4
Ground Threat Detection Scenario

Initially, the scenario was run with the SA-9 weapons on hold. The lead flight aircraft detected the SA-9 site at about four miles out (consistent with the input visual range parameters) and continued en route with no evasive maneuvers. The trail flight followed the exact same route.

The scenario was repeated with the SA-9 site on weapons free. Again the F-16Ds detected the SA-9, but continued en route and were shot down by the SA-9s. The trail flight continued and were also shot by the SA-9s with no evasive maneuvering.

Bingo Fuel

An F-14D entity was used to test ModSAF's bingo fuel task. The F-14D was assigned a sweep task with 2,500 lbs of fuel. The status page indicated that 2,366 lbs of fuel were required to return to base (RTB) for the selected bingo fuel point. The F-14D flew over 80 nm and the status page never indicated a decrease in fuel. Switching over to the entity creation page showed that the fuel had decreased to 2,273 lbs, 93 lbs less than needed to RTB. The F-14D continued on the sweep mission and would transition to RTB only after the operator manually entered a lower value of onboard fuel on the entity creation page while the aircraft was actually flying the sweep.

Winchester Weapons

The F-14D was also used to test whether or not the aircraft would correctly RTB if it ran out of weapons during a sweep mission. The F-14D was initialized with only two Sidewinders. Five MiG-29s were flown head-on to the F-14D. The F-14D fired both its missiles at the MiG-29s. Although the F-14D had no remaining weapons and was aware of the MiG-29 threats on its nose, it continued its sweep making no attempt to RTB. The test was repeated with the F-14D initialized with no weapons and produced the same results.

Terrain Following

Terrain following was tested by assigning a ModSAF F-16D ingress task over a section of the National Training Center database. The terrain varied from 1,800 ft to over 6,000 ft along the flight path. The model varied altitude along the route and avoided the terrain. Climb rates, terrain clearance, and G loading along the route were not analyzed.

EVALUATION OF MODSAF FLIGHT SOFTWARE

Some of the fwa_ and hm_ software files were reviewed along with two pieces of documentation from the Programmer's Reference Manual and the on-line monitor displays of parameter default initial values. The documentation reviewed consisted of seven pages of parameter initialization (US_F-16D parameters) and a condensed single-page overview of LibFwa with many obvious typographical errors causing equation errors.

An examination of the documentation and code suggests that it is based on a limited understanding of aerodynamics and mechanics. The code is almost impossible to read or maintain because many variables are given names or definitions that are inconsistent with their application in the algorithms, the comments are unilluminating, and the units are rarely defined. In addition, the effects of many of the algorithms are inconsistent with the physics of the real

world. The sparse top-level documentation does not always match the code and is insufficient to describe the architecture or functions. A few examples of these problems are presented in the following sections.

Axis Systems, Angles, and Hull_to_World Matrix

Function fwa_tick in file fwa_tick.c calculates a matrix called hull_to_world. This is calculated in terms of the trigonometric functions of the roll, track, and flight path (climb) angles. Angle of attack (AOA) is used in the calculation of many flight characteristics, as described later, yet no account is taken of its effect on the attitude of the hull relative to the flight path and thus relative to earth. Does the omission of AOA from this matrix mean that although the effect is accounted for in aircraft performance, it is not to be represented in the aircraft attitude?

Analysis of much of the software suggests that roll, flight path angle, and track have their conventional definitions with respect to map Grid North, Grid East, and Down. Analysis based upon this shows that the matrix is world_to_hull, not hull_to_world. Furthermore, the world axes are Grid North, Grid East, and Up and the hull axes are Left, Forward, and Top. Both sets of axes represent unconventional and left-handed systems. Relative to these axes, roll, flight path angle, and track are not Euler angles. This matrix is used in many places, but its use was not evaluated.

Immediately after the matrix calculation, the X, Y, and Z components of earth axis velocity are calculated from the speed and trigonometric functions of flight path angle and track. The equations used confirm that the world axes correspond to Grid North, Grid East, and Up. Further confirmation of the angle and axis conventions is provided elsewhere by track corresponding to atan2(Y_vel, X_vel).

Even with knowledge of the unconventional axis systems, to use a standard matrix vector multiply function to obtain the world axis velocities would be wrong, since the matrix is named as hull_to_world, the inverse of its real meaning.

The transformation between map grid and geodetic coordinates has not been analyzed with respect to the suitability of the map projection or proper accounting for meridian convergence when transforming heading, velocity, and acceleration between systems.

Equations Of Motion

Track, flight path angle (climb), and roll angle are integrated from their individual rates of change, a poor method especially near the vertical. The roll, track, and flight path angles are each expressed in the range $-\pi$ to $+\pi$ through the application of the function angle_clip. This constraint is proper for roll and track but is totally wrong for pitch angle which should never

exceed $\pi/2$. The integration of pitch angle near the vertical needs special care, and it is hard to visualize the functioning of the simulation at a declared pitch angle value of 120^0.

Speed by definition is along the flight path and is integrated from speed rate.

Sideforce And Sideslip

In the code, the contributions of sideforce to fpa->rate and track->rate have the wrong sign unless sideforce is defined as positive to the left, although it is consistent with what we learned about the axes through analysis of the hull_to_world matrix. The documentation of LibFwa shows sideforce in the flight_path_angle_rate calculation, but it is omitted from the track_rate.

Although sideforce affects the attitude rate, the contribution is ignored in much of the code, as when it calculates desired forces, maximum turn performance, limits, etc. The calculation of sideforce is not shown in the documentation, nor was it found in the code. Sideslip, which might be expected in the presence of sideforce, is not modeled. On those rare occasions when a pilot might apply sideforce and sideslip, a complicated aerodynamic and artificial intelligence situation is produced that requires an extremely high level of sophistication for an unmanned simulation. The rarity of the application of sideforce and the difficulty of appropriately modeling it suggests that there is little reason to include sideforce in CGF.

Second Order Control Of Angle Of Attack

An angle of attack goal is calculated from the desired lift. Angle of attack and thus lift is controlled as a second-order system with a damping ratio of 0.8. The comments mention damping ratio Eta, whereas the conventional name is Zeta. More importantly, complicated difference equations are implemented. Since the selected properties of the system are arbitrary, it seems more reasonable to use simple and understandable difference equations that give a very close approximation to the correct solution provided the selected natural frequency is slow compared with the computation interval.

Second Order Roll Control And Effect Of Limits

Roll angle is controlled toward the roll goal with second-order equations similar to those used for angle of attack, but this time the system has critical damping. Roll->actual is integrated from the roll->desired_rate, which itself is set after the angle integration.

Roll rate is not explicitly limited in this function. However, function fwa_limits contains the line:

roll->rate = roll->desired_rate, limited between +/-π.

Since the value of roll->rate does not affect roll->desired_rate, the function has no effect on the second order control of roll angle.

An additional problem with the roll rate limits is that they are set constant at $-\pi$ to $+\pi$. Although these are reasonable values for unsophisticated limiting of body axis roll rate, the roll angle being integrated is the roll relative to earth. This may have a large rate of change near the vertical even with a low body axis rate. Appropriate limits for the rate of change of roll angle need to be calculated and applied in real time, or the whole attitude integration system should be changed.

Max Turn Rate and Effect of Limits on Rates of Change of Track and Flight Path Angle

Function get_turn_performance sets
normal accel = max aero lift / mass.

Then it sets,
max turn = normal accel/speed.

No structural loading limits are applied nor is account taken of the resolution of thrust through angle of attack or of the possible application of sideforce, although it is modeled elsewhere. In addition, the fundamental relationship to normal force is wrong. As occurs with much of the limit code, there is confusion between loading normal to the flight path and acceleration; the effect of gravity on the flight path is ignored. What the code calls normal acceleration is, in fact, normal loading in units of acceleration. The turn performance relation should be:

$$\text{max\_level\_turn} = \text{sqrt}(\text{normal\_loading}^2 - G^2)/\text{speed}; \quad \text{(if real)}$$

In function fwa_control, the desired track rate is derived from the track error, then bounded in terms of the current maximum lift with the implied assumption that all the loading or lift is available for turn acceleration in the horizontal plane. However, in practice, some of the lift is required for acceleration or equilibrium in the vertical direction. Similarly, the desired flight path rate is bounded without reference to turn requirements or the need to oppose gravity. No account is taken of the possible application of sideforce.

Track_force and fpa_force are calculated from the improperly limited desired track and flight path angle rates. The roll->goal is calculated from these forces after further adjustment and limiting through the total normal force. The roll goal ensures that the (limited) desired flight path rate is achieved, and what lift remains is used to generate a modified desired track rate. Since the calculation started with incorrect assumptions, it produces an incorrect answer.

Much of the trouble with the application of the limits results from starting with separate desired track and flight path angular rates. If it started with a single desired normal acceleration, the application of limits would be simpler.

In the above procedure, a further error was found with the calculation of the flight path angle force. This force is normal to the velocity and in the vertical plane containing the velocity. It is calculated as:

fpa_force = mass * speed * cos(roll->actual) * fpa_desired rate + mass * G * cos(fpa->actual)

The cos(roll->actual) should not be in this equation.

Properties of the Atmosphere

File hm_utils.c contains four functions representing properties of the ICAO standard day:

hm_air_density(altitude)
hm_air_density_rat_sq(altitude)
hm_mach_velocity(mach, altitude)
hm_velocity_mach(velocity, altitude)

The last two functions are correctly and adequately described in the comments and make a good approximation to converting speed, in m/sec, to mach or the reverse, as a function of altitude in meters. This is very reasonable, since ModSAF is mainly a metric simulation.

The first two functions generate reasonable approximations to ambient density ratio and its square, respectively. The comments do not describe the functions or their units. The first function name suggests that it calculates actual air density (ρ) rather than what it does calculate, density ratio (σ), which is the ratio of ρ to the standard value at sea level, ρ_0. However, the name of the second function correctly leads to the expectation that it calculates σ^2.

Numerical analysis of both of these density related functions shows that for the output to be correct, the input altitude must be in feet despite this being a metric simulation.

Function fwa_tick(...) in file fwa_tick.c contains the following line:

fwa->air_density = hm_air_density(position[Z]);

The above is both misleading and wrong. Position[Z] is in meters, yet the function needs feet as an input, as described earlier. Thus the function output is seriously wrong. Also, the function calculates density ratio, but the value is assigned to something called air density. This error of calculating density ratio (σ) but assigning it to something called air density or rho occurs throughout the simulation.

The one page overview of LibFwa states:

air_density = initial_air_density * (the expression for density ratio, σ),
where initial_air_density = .0249

Thus, unlike the code, the documentation does include ρ_0, but in what units? The correct value is .00237688 slug/ft^3 or 1.225 kg/m^3 in English or metric units, respectively.

Lift, Lift Coefficient, and Lift Curve Slope

Function fwa_control generates an aerodynamic lift limit which is based on an angle of attack limit α_{limit}. This angle of attack limit must be in radians to be consistent with other lift and drag software. With some nomenclature simplification, the lift limit is set by:

corr_lift_max = .5 * speed^2 * fwa->air_density * fwa->params->lift_coef * α_{limit}

Rearrangements of the above relationship occur throughout the software.

It has already been shown that fwa->air_density is, in fact, the density ratio (σ). Therefore, for the above expression to be dimensionally and physically correct, the variable fwa->params->lift_coef must represent the product of wing area (S), air density at sea level (ρ_0), and lift curve slope (CL_α):

fwa->params->lift_coeff = S * CL_α * ρ_0.

For an aircraft like the F-16, the lift curve slope should be around 3.5; for no aircraft should it be greater than 2π. Knowing that the F-16 wing area is 27.8709 m^2 (300 ft^2) and that the sea level air density is 1.225 kg/m^3, we can calculate the simulated lift curve slope from the above expression once we know the value of fwa->params->lift_coef. An on-line initialization page shows that a coefficient of lift is set at 404.13, and this value is also shown in the parameter value documentation. This yields a CL_α value of 11.8, which is several times larger than is realistic.

The documentation overview of LibFwa gives a different relationship that may be summarized as:

Coef_lift = 21522.3
initial_air_density = .0249
lift = .5 *σ * initial_air_density * speed^2 * coef_lift * AOA

Substituting the real-world relation:

$lift = .5 * \sigma * \rho_0 * speed^2 * S * C_{L\alpha} * AOA$

into the above gives:
$coef\_lift = S * C_{L\alpha} * \rho_0 / initial\_air\_density$

Substituting metric numerical values into the above and rearranging gives: $C_{L\alpha} = .0249 * 21522.3 / (1.225 * 27.8709) = 15.696$ per radian. This value is 33% larger than the value derived by analyzing the code. Both values are several times larger than reasonable.

Although the meaning of coef_lift differs between the code and the LibFwa documentation, in neither case does it correspond to what aerospace engineers mean by lift coefficient, C_L, which is defined as $lift/0.5\rho V^2 S$.

Drag Due to Lift

The drag due to lift model in function fwa_drag is based on the assumption that the resultant force due to lift acts close to the aircraft Z direction. Because the oversized lift curve slope yields too small an AOA, this model gives an induced drag value that is many times too small.

Even if the AOA were properly calculated, this would be a poorly induced drag model. A much better approach is to calculate CD_i as a function of C_L specific for each aircraft type.

Drag at Zero Lift

The zero lift drag model in function fwa_drag corresponds to:

coef_drag = fwa->params->missile_coeff + fwa->params->airplane_drag_index.

Subsonic drag is:

$drag = fwa->air\_density * V^2 * coef\_drag / 2;$

An analysis similar to that for lift, remembering the air density term actually is density ratio, σ, shows that:

$coef\_drag = S * C_D * \rho_0$

The parameter documentation states that the drag index is set to .83313, which is therefore the value of coef_drag without missiles. From the above relationship, this corresponds to:

$C_D = .83313 / (1.225 * 27.8709) = .0244.$

Such a value is believable at low Mach numbers but is of unknown accuracy.

Supersonic zero lift drag is calculated as 1.06 * the subsonic value. This is poor. The model should be a continuous function of Mach number.

The meaning of coef_drag does not correspond to what aerospace engineers mean by drag coefficient, C_D, which is defined as drag/$0.5\rho V^2 S$.

Thrust

Function hm_thrust calculates maximum thrust for a given flight condition, reading the value from a table of thrust versus speed and altitude.

If speed and altitude are off the table, the function multiplies the first thrust value in the table by σ^2. The square is a poor thing to do. If this square was the only reason for having a function that generates σ^2, the function hm_air_density_rat_sq is not needed. The comments claim that this calculation gives the minimum thrust necessary to counteract drag, yet drag was not considered nor does it appear relevant.

The comments claim that little would be gained by interpolating on the table, so the nearest value of thrust in the table is used. Even though the data may be poor, it would be preferable to interpolate between data points to ensure maximum thrust changes smoothly with flight conditions.

It would be preferable to tabulate thrust/σ rather than thrust, because the value would change much slower with altitude, permitting fewer data points. It would also be preferable to follow standard conventions and use Mach Number rather than speed for the second independent variable.

Fuel Mass Units and Density

Function fwa_get_current_mass has the following single executable line of code.

```
return(fwa->params->vehicle_mass + fwa->fuel * fwa->ppl);
```

Is the ppl pounds per liter? Is fuel in liters? Is mass in kg? An initialization page shows fuel density set at 6.8 pounds per gallon, but what is the value of fwa->ppl?

Observations Based Upon Documentation Of US_F16D_params.rdr

Extracts from the documentation of F-16D parameters, US_F16D_params.rdr, are given below with additional comments.

(airplane_drag_index .83313) ;; 132.4 (airp_drag_index) * 0.0249 air_dens_msl

Once more, .0249 is not the correct value for ρ_0. The product should equal 3.29676, nearly four times greater than the value assigned. What actually got into the code? The physical meaning of the term needs to be explained.

(vehicle_mass 9,100.00) ;; kg . This is 20,062 lb, excluding fuel and stores.

(lift_min -177,956.0) ;; N, corresponds to -2G structural limit.

(lift_max 800,803.0) ;; N, corresponds to +9G structural limit.

The above structural lift limits correspond to the stated G limits at the very low empty weight. Thus, if applied, they seriously curtail maneuverability at higher weights, regardless of the available aerodynamic lift. It would be better to specify structural limits in G, possibly as a function of stores, and calculate the corresponding lift limits in real time as a function of weight.

There was no reference to limits on Mach or equivalent airspeed.

Absence of Function to Perform Linear Function Interpolation

Function fwa_fuel_usage_rate calculates fuel flow as a function of altitude and percent engine performance by linear interpolation on a two-dimensional table of data. The process is coded specifically for this fuel flow function with frequent references by name to the input variables and properties of the referenced data table. The process does not use a general purpose function for performing Linear Function Interpolation (LFI).

As described earlier, function hm_thrust calculates maximum thrust from a two-dimensional data table without using linear interpolation; it uses the nearest value in the table. Like the fuel flow, this process is coded with specific reference to the names of the input variables and data table.

Taken together, these two pieces of information strongly suggest that there is no general purpose LFI function. If there had been a general purpose function, it would have been used on the fuel flow and thrust; there would be no need for comments explaining why interpolation was not performed for thrust. It also explains why many other properties are over-simplified, rather than being modeled as LFIs.

General Observations on Flight Model

The preceding sections have addressed just a few of the problems in the flight and control model software algorithms. Most of the other software examined has shown a low level of expertise in the algorithms similar to that highlighted herein. Lack of the usual axis or sign conventions, poor math and physics, misuse of nomenclature, carelessness with units, and

inadequate documentation make the aerodynamic software hard to understand, repair, maintain, or use.

In some respects, the code has been over-modularized, with similar relationships occurring in many modules. For instance, the lift relationship or its inverse is repeated or implied in many functions, all of which must be modified if the lift relationship is changed. Some functions use or imply relationships slightly different from others, which may cause errors. For instance, a stall speed equation ignores the contribution of thrust resolved through the angle of attack. Another example is that some functions use sines and cosines when resolving thrust through the AOA, whereas others use small angle assumptions. As already mentioned, the effect of sideforce is included in some functions but not in others.

MODSAF ARCHITECTURE

This section evaluates ModSAF architecture based on the software design methodology, without regard for the specific implementation of aerodynamic and flight algorithms described above. The "Software Architecture and Overview Document for ModSAF[1]" distributed with ModSAF details the design methodology. The document describes four techniques used in the development of ModSAF:

Layering
Object-based Programming
Rigorous Interface Specification
Data Driven Execution

These techniques provide the basis for a sound software development strategy. Adherence to these techniques eases the management of parallel software development and test activities. The resulting modular software facilitates growth and modification of libraries for future ModSAF releases as well as user enhancements for specific site requirements. Data driven execution provides the means to add new vehicle types with minimal software changes. New entities may be added through the modification of parametric data files.

To take advantage of data driven execution, the underlying ModSAF algorithms and parametric data must be capable of fully supporting the entities' performance and behavioral characteristics. As noted in Section 4.0, the algorithms and parametric data do not accurately depict fixed-wing aircraft and make it difficult to add new aircraft types.

The latest distribution of ModSAF, version 2.0, contains 394 software libraries with over 600,000 lines of C source code. The software is intended to be portable to most Unix workstations supporting X windows and Motif for the Graphical User Interface (GUI) operation. The software is also intended to be operating system independent by selecting the correct

[1] Software Architecture and Overview Document for ModSAF, Version 2.0, Sep 29, 1995

platform and operating system options for compilation. Although the system has been run on many different systems, portability is an issue based on ModSAF reflector messages.

ModSAF is a real-time, entity-level simulation. Each simulated entity's state is updated periodically using a variable period (or tick). During low levels of activity, ModSAF updates each entity every 67 msec. As the activity increases, the ModSAF update period lengthens to accommodate all locally simulated and remotely generated entities within each processing period. This allows the system to handle spikes in processing requirements. The software allows the user to fully utilize the available processing capability on the ModSAF host based on a nominal load. If spikes in processor requirements occur, the system gracefully handles the increased requirement by slowing down until the requirements again normalize.

The number of entities simulated, network traffic, types of entities, activity of the entities, and surrounding terrain are some of the factors that affect update rate. Also, the required update rate for entities varies based on the use of the simulation.

ModSAF benchmark tests are run using the criterion that each entity over the course of one minute must, 90% of the time, be updated every 500 msec. Using this criterion, the developers determine the maximum number of active ground entities a specific version of ModSAF can simulate. This gives them a basis for determining how software changes impact processing requirements at the system level. It also provides a method for determining run-time efficiency across hardware platforms.

Although this criterion is adequate for engineering comparison and may be suitable for ground force interaction, much faster update rates are required to generate air entities flying with or engaging in a human-in-the-loop virtual simulation. As the system slows down, entities start jumping, making it difficult for a manned simulator to engage air entities within visual range. As the system slows down further, the manned simulation begins to lose radar lock during beyond visual range engagements.

A test was run using an SGI Indigo2 with a MIPS R4400/200 MHz processor with 128 MB of memory as the SAFsim. An SGI Indy was used with a MIPS 4600/134 MHz processor and 96 MB of memory as the SAFstation. Both machines were running IRIX 5.3 operating system. With no network traffic, no engagements, ModSAF F-16Ds flying ingress routes began slowing the system down at about 20 entities. At 60 entities, the system update rate had doubled.

When running ModSAF during large-scale DIS exercises such as Synthetic Theater of War-Europe (STOW-E) and Warfighter 95, the system slows down significantly just processing network entities. Additional tests are required to determine how the network traffic reduces the number of air entities ModSAF can simulate. In conjunction with these tests, it is also necessary to identify the update rate where the simulation becomes unacceptable for interaction with manned flight simulators.

ModSAF is capable of distributing the processing load across workstations by using the Persistent Object (PO) Protocol. In addition to the DIS protocol, ModSAF workstations on the same PO database number are able to share simulation tasks across the same physical network used for DIS traffic. The system is also able to pick up the simulation tasks of another workstation should it fail. One drawback to the PO protocol is the additional traffic generated on the network. Although the PO protocol uses a different User Datagram Protocol (UDP) port number than the DIS traffic, the traffic is broadcast to all machines participating on the network. This broadcast usually does not create problems for local area networks. It is, however, a consideration in wide area networks because gateways, encryption devices, and the leased commercial lines restrict the total bandwidth available for DIS exercises.

RESULTS

This evaluation indicates that the current version of ModSAF cannot provide air entities capable of effectively interoperating with human-in-the-loop flight simulators. Although the evaluation focused primarily on the F-16D, analysis of the general fixed-wing aircraft software suggests the same general problems with all fixed-wing aircraft entities.

In general, the software architecture is sound and well organized by the ModSAF developers. Unfortunately, the good software architecture forces dependencies on the common fixed-wing aircraft libraries that are incorrect and poorly documented. Therefore, incorporating a correctly modeled air entity cannot be accomplished without first redesigning the existing fixed-wing aircraft, hulls, and other code discussed previously. The problems are too big to be fixed with a few patches.

It appears that the air entity simulations were developed without sufficient aid from aerospace or mechanical engineers. As a result, incorrect algorithms were used, in some cases, to model the aerodynamics and flight mechanics of air entities. In addition, failure to use standard aerospace and simulation engineering conventions has resulted in models that are inefficient and difficult to maintain.

In addition, the following list suggests that the ModSAF F-16 simulation was developed without appropriate subject matter expertise:

F-16D simulation with no F-16A or F-16C modeled
No F-16 air-to-air capability
No two-/four-ship air-to-air capability for entities with an air-to-air mission
Limited/incorrect weapon selections
Threat detection routines do not consider radar warning receivers (RWR)
Threat detection routines do not consider visual detection for air-to-air missions
No command and control at any level modeled for air entities
No threat avoidance capability; air entities blindly follow route

The Graphical User Interface provides an easy interface for creating entities and setting up basic routes. The user's manual is difficult to follow while assigning tasks. All tasks are grouped under fixed-wing aircraft task frames, but not all aircraft are capable of correctly carrying out all tasks. Some can be assigned to aircraft with a visual sensor only, others to individual entities only, not to units.

No documentation of the fixed-wing aircraft configuration menu could be found. The majority of the parameters in this menu should not be available to the operator for modification. Lift coefficient, thrust map, take-off speed, maximum AOA, etc. should be calculated in the flight equations based on the aircraft's aerodynamic properties and the stores selected by the operator.

RECOMMENDATIONS

To use ModSAF as the CGF system to depict air entities that will be interacting with human-in-the-loop, fixed-wing aircraft simulators during ADS exercises, the following recommendations are provided:

1. Conduct a test to determine if any ModSAF platform can provide an update rate adequate for interaction with a manned simulator while processing the large number of network entities anticipated during STOW 97.

2. Completely redesign the fixed-wing aircraft aerodynamic and flight dynamic algorithms and data files using engineers with backgrounds in these subjects.

3. Develop an F-16C model with complete air-to-air and air-to-ground sensor capability. Include AIM-120, CBU-87, GBU10/12, Mk-84 and 20 mm gun weapons. Use subject matter experts to define F-16C weapon system behaviors.

4. Clearly identify which version of munitions are being simulated (e.g., AGM-65C).

5. Allow the user to assign the appropriate mission to the aircraft based on scenario requirements.

6. Provide fixed-wing aircraft threat detection and Identification Friend or Foe (IFF) capabilities consistent with actual aircraft capabilities.

7. Add the capability for air-to-air tasks to be conducted as a two- or four-ship unit.

8. Add a Headquarters or Airborne Warning and Control System (AWACS)-type command and control logic for both Blue and opposition force air.